

## Tarea Practica (0307): Sistema de Recomendación

(Universidad de Guanajuato, División de ingenierías campus Irapuato –  
Salamanca, [j.chavezislas@ugto.mx](mailto:j.chavezislas@ugto.mx), Inteligencia Artificial)

### INTRODUCCION

En esta tarea se implementará un sistema de recomendación a partir de un código ya hecho por el profesor y visto en clase llamado: “syla\_Clustering\_ART1.cpp” el cual nos da un resultado de 22 personas (imagen1) capturando sus gustos en cuanto a un catálogo de 10 géneros de películas (Imagen2).

```
The following 22 SAMPLES will be grouped:
Sample 0: 1 0 0 0 0 1 1 0 1 1
Sample 1: 1 0 1 1 0 0 1 0 1 1
Sample 2: 1 0 1 1 1 0 0 1 1 0
Sample 3: 1 0 1 1 1 1 0 1 1 1
Sample 4: 1 0 1 1 0 1 1 1 0 0
Sample 5: 1 1 1 1 1 1 1 1 1 1
Sample 6: 1 0 1 0 1 1 1 1 0 1
Sample 7: 1 1 1 0 1 1 1 1 0 0
Sample 8: 1 0 0 1 1 1 1 0 0 0
Sample 9: 1 0 0 1 0 0 1 1 1 0
Sample 10: 1 1 1 1 1 1 0 0 1 0
Sample 11: 1 0 0 1 1 0 0 0 1 1
Sample 12: 1 1 1 0 1 1 1 0 1 1
Sample 13: 0 0 0 1 0 1 1 0 0 1
Sample 14: 0 1 1 1 0 0 1 1 1 1
Sample 15: 1 0 1 0 1 1 1 0 1 1
Sample 16: 1 0 1 0 0 1 1 0 0 1
Sample 17: 1 0 1 1 1 1 1 0 1 0
Sample 18: 1 1 0 1 1 1 1 0 1 1
Sample 19: 1 1 0 1 0 1 1 0 0 1
Sample 20: 1 1 1 1 0 0 1 0 1 1
Sample 21: 1 0 0 0 1 0 1 0 1 0
```

Imagen1

```
Example of using the ART1 algorithm.
10 FEATURES (binary) are used in each row-ordered sample:
Feature 1: Action - Adventure
Feature 2: Biography - History
Feature 3: Comedy
Feature 4: Horror
Feature 5: Drama
Feature 6: Romance
Feature 7: Sci-Fi - Fantasy
Feature 8: Sport
Feature 9: Thriller - Crime
Feature 10: War and Western
```

Imagen 2

Una vez teniendo los gustos de las personas, se agrupan las personas mediante sus gustos en común (Imagen3).

```
Finally, 8 clusters were found.

Cluster 1 0 0 0 0 0 1 0 1 0
includes
Sample 0: 1 0 0 0 0 1 1 0 1 1
Sample 1: 1 0 1 1 0 0 1 0 1 1
Sample 9: 1 0 0 1 0 0 1 1 1 0
Sample 21: 1 0 0 0 1 0 1 0 1 0

Cluster 1 0 0 1 1 0 0 0 1 0
includes
Sample 2: 1 0 1 1 1 0 0 1 1 0
Sample 3: 1 0 1 1 1 1 0 1 1 1
Sample 10: 1 1 1 1 1 1 0 0 1 0
Sample 11: 1 0 0 1 1 0 0 0 1 1
Sample 17: 1 0 1 1 1 1 1 0 1 0

Cluster 1 0 1 0 0 1 1 0 0 0
includes
Sample 4: 1 0 1 1 0 1 1 1 0 0
Sample 6: 1 0 1 0 1 1 1 1 0 1
Sample 7: 1 1 1 0 1 1 1 1 0 0
Sample 16: 1 0 1 0 0 1 1 0 0 1

Cluster 1 1 1 1 1 1 1 1 1 1
includes
Sample 5: 1 1 1 1 1 1 1 1 1 1

Cluster 0 0 0 1 0 1 1 0 0 0
includes
Sample 8: 1 0 0 1 1 1 1 0 0 0
Sample 13: 0 0 0 1 0 1 1 0 1 0
Sample 19: 1 1 0 1 0 1 1 0 0 1

Cluster 1 0 1 0 1 1 0 0 1 1
includes
Sample 12: 1 1 1 0 1 1 0 1 1 1
Sample 15: 1 0 1 0 1 1 1 0 1 1

Cluster 0 1 1 1 0 0 1 0 1 1
includes
Sample 14: 0 1 1 1 0 0 1 1 1 1
Sample 20: 1 1 1 1 0 0 1 0 1 1

Cluster 1 1 0 1 1 1 1 0 1 1
includes
Sample 18: 1 1 0 1 1 1 1 0 1 1
```

Imagen 3

Entonces una vez acomodados en grupos se nos sugiere lo siguiente:

1. Mejora el programa compartido en clase o desarrolla tu propia versión en Python o cualquier otro lenguaje de tu preferencia verificando que,
2. (7 puntos) después de haber incluido a todos los vectores dentro de alguna clase, sigan manteniéndose las condiciones de inclusión una vez que los prototipos pudieron ser modificado al incluir otros elementos.
3. Asegúrate de mostrar adecuadamente las agrupaciones obtenidas (como en el ejemplo) y agrega una etapa de recomendación, sujeta a lo siguiente:
4. (3 puntos) Sólo se podrá emitir una recomendación para los casos de agrupaciones con al menos 4 elementos, o bien
5. Se emitirá una recomendación del tipo “El sujeto X podría considerar la característica Y” debido a que TODOS LOS OTROS miembros de su agrupación sí la consideran.
6. De modo opcional, podrás mejorar la interfaz para la definición de las características, para la posible lectura de las características desde un archivo, etc.

## METODOLOGIA

¿Cómo podríamos llegar a esta solución?

¿Cómo podemos complementar este programa?

Para el desarrollo de esta nueva implementación, consideré primero juntar los atributos y las funciones del programa original en una clase tal y como se pedía en la consigna, y una vez teniéndolo de esa manera de forma funcional, se puede agregar una función la cual se encarga de generar las recomendaciones tomando en cuenta lo siguiente:

Recorre cada uno de los prototipos (clusters) generados. Para cada prototipo, calcula el tamaño del cluster contando cuántos elementos pertenecen a él.

Si el tamaño del cluster es mayor o igual a 4, comienza a generar recomendaciones para ese cluster. Si no, pasa al siguiente prototipo.

Para generar recomendaciones, recorre cada elemento en el cluster. Para cada elemento, recorre cada característica del elemento.

Si el elemento no tiene una característica (es decir, el valor de la característica es 0), verifica si todos los demás elementos en el cluster tienen esa característica.

Si todos los demás elementos en el cluster tienen la característica, emite una recomendación de que el elemento actual podría considerar tener esa característica, ya que todos los demás miembros en el cluster la tienen.

```
#include <iostream>

using namespace std;

const int NSAMP = 22;
const int NFEAT = 10;
```

```
const int inputs[NSAMP][NFEAT+1]
=
{
    {1,0,0,0,0,1,1,0,1,1,-1}, //
    Sample 0

    {1,0,1,1,0,0,1,0,1,1,-1}, //
    Sample 1

    {1,0,1,1,1,0,0,1,1,0,-1}, //
    Sample 2

    {1,0,1,1,1,1,0,1,1,1,-1}, //
    Sample 3

    {1,0,1,1,0,1,1,1,0,0,-1}, //
    Sample 4

    {1,1,1,1,1,1,1,1,1,1,-1}, //
    Sample 5

    {1,0,1,0,1,1,1,1,0,1,-1}, //
    Sample 6

    {1,1,1,0,1,1,1,1,0,0,-1}, //
    Sample 7

    {1,0,0,1,1,1,1,0,0,0,-1}, //
    Sample 8

    {1,0,0,1,0,0,1,1,1,0,-1}, //
    Sample 9

    {1,1,1,1,1,1,0,0,1,0,-1}, //
    Sample 10

    {1,0,0,1,1,0,0,0,1,1,-1}, //
    Sample 11

    {1,1,1,0,1,1,0,1,1,1,-1}, //
    Sample 12

    {0,0,0,1,0,1,1,0,0,1,-1}, //
    Sample 13

    {0,1,1,1,0,0,1,1,1,1,-1}, //
    Sample 14

    {1,0,1,0,1,1,1,0,1,1,-1}, //
    Sample 15

    {1,0,1,0,0,1,1,0,0,1,-1}, //
```

```

Sample 16
        {1,0,
1,1,1,1,1,0,1,0,-1}, // Sample 17

{1,1,0,1,1,1,1,0,1,1,-1}, //
Sample 18

{1,1,0,1,0,1,1,0,0,1,-1}, //
Sample 19

{1,1,1,1,0,0,1,0,1,1,-1}, //
Sample 20

{1,0,0,0,1,0,1,0,1,0,-1} //
Sample 21
        };

class ART1
{
    int **E;
    int **P;
    int nPrototypes;
    float beta;
    float rho;

public:
    ART1(float beta, float rho):
    beta(beta), rho(rho) {
        E = new int*[NSAMP];
        for(int i = 0; i < NSAMP;
        ++i)
            E[i] = new
int[NFEAT+1] ();

        for (int i = 0; i <
NSAMP; ++i)
            for (int j = 0; j <
NFEAT+1; ++j)
                E[i][j] =
inputs[i][j];

        P = syMatrixNew(NSAMP,
NFEAT);
        nPrototypes = 0;

syVectorCopyTo(E[0],P[0],NFEAT);
        E[0][NFEAT] = 0;
        nPrototypes++;
    }

    int syProximityTest(int
    *E,int *P, int nD, float beta)
    {
        int vAnd[25];
        syVectorAnd(E,P,vAnd,nD);
        int mag =

```

```

syVectorMagnitude(vAnd,nD);

        float termL =
mag/(beta+syVectorMagnitude(P,nD)
);

        float termR =
syVectorMagnitude(E,nD)/(beta+nD)
;

        if (termL > termR) return
1;

        return 0;
    }

    int syVigilanceTest(int
    *E,int *P, int nD, float rho)
    {
        int vAnd[25];
        syVectorAnd(E,P,vAnd,nD);
        int mag =
syVectorMagnitude(vAnd,nD);

        float termL = (float)
mag/syVectorMagnitude(E,nD);

        if (termL >= rho) return
1;

        return 0;
    }

    int syVectorMagnitude(int *v,
int nD)
    {
        int mag = 0;
        for (int k=0; k<nD; k++)
            mag += v[k];
        return mag;
    }

    int syVectorAnd(int *v1, int
    *v2, int *vA, int nD)
    {
        for (int k=0; k<nD; k++)
            if(v1[k]==1 &&
v2[k]==1) vA[k]=1;
            else vA[k]=0;
        return 0;
    }

    int syVectorCopyTo(int *S,
int *D, int nD)
    {
        for (int k=0; k<nD; k++)
            D[k] = S[k];
        return 0;
    }

```

```

    int syVectorMergeInProto(int
    *E, int *P, int nD)
    {
        for (int k=0; k<nD; k++)
            if (E[k]==0 ||
P[k]==0) P[k]=0;

        return 0;
    }

    int syVectorDisplay(int *v,
int nD)
    {
        for (int k=0; k<nD; k++)
            cout << v[k] << " ";
        cout << "\n";
        return 0;
    }

    int **syMatrixNew(int nR, int
nC)
    {
        int **M;
        M = new int *[nR];

        for (int k=0; k<nR; k++)
            M[k] = new int[nC];

        for (int r=0; r<nR; r++)
            for (int c=0; c<nC;
c++)
                M[r][c] = 0;

        return M;
    }

    int syMatrixDelete(int **M,
int nR)
    {
        for (int r=0; r<nR; r++)
            delete[] M[r];
        delete[] M;
        return 0;
    }

    void
generateRecommendations()
    {
        for (int r=0;
r<nPrototypes; r++)
        {
            int clusterSize = 0;
            for (int s=0;
s<NSAMP; s++)
                if (E[s][NFEAT]
== r)

```

```

clusterSize++;

                if (clusterSize >= 4)
                {
                    cout <<
"\nCluster " << r << " has " <<
clusterSize << " elements.
Generating recommendations...\n";
                    for (int s=0;
s<NSAMP; s++)
                    {
                        if
(E[s][NFEAT] == r)
                        {
                            for (int
f=0; f<NFEAT; f++)
                            {
                                if
(E[s][f] == 0)
                                {
                                    bool allOthersHaveFeature = true;
                                    for (int t=0; t<NSAMP; t++)
                                    {
                                        if (E[t][NFEAT] == r && t != s &&
E[t][f] == 0)
                                        {
                                            allOthersHaveFeature = false;
                                            break;
                                        }
                                    }

                                    if (allOthersHaveFeature)
                                    {
                                        cout << "Sample " << s << " might
consider feature " << f << " as
all other members in the cluster
have it.\n";
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }

    void run()
    {
        for (int m=0; m<NSAMP;

```

```

m++)
    {
        int merged = 0;
        for (int n=0;
n<nPrototypes; n++)
        {
            if
(syProximityTest(E[m],P[n],NFEAT,
beta) &&
syVigilanceTest(E[m],P[n],NFEAT,
rho) )
            {

syVectorMergeInProto(E[m], P[n],
NFEAT) ;

                E[m][NFEAT] =
n;

                merged = 1;
                break;
            }
        }
        if (!merged &&
nPrototypes < NSAMP) // Asegúrate
de que nPrototypes es menor que
NSAMP
        {

syVectorCopyTo(E[m],P[nPrototypes
],NFEAT);

                E[m][NFEAT] =
nPrototypes;

                nPrototypes++;
        }

        cout << "\n\n\nFinally,
"<nPrototypes<< " clusters were
found.\n";
        for (int r=0;
r<nPrototypes; r++)
        {
            cout << "\nCluster
";

syVectorDisplay(P[r],NFEAT);
            cout << "includes\n";
            for (int s=0;
s<NSAMP; s++)
                if (E[s][NFEAT]
== r)
                {

printf("Sample %2d: ",s);

syVectorDisplay(E[s],NFEAT);

                }

```

```

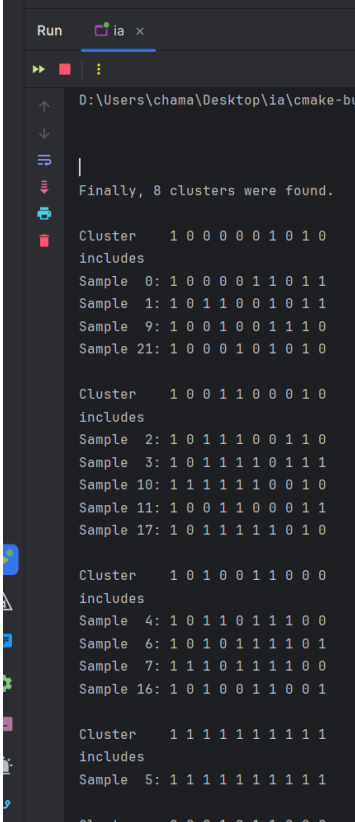
generateRecommendations();

        syMatrixDelete(P,NSAMP);
        cout << "\n\nTo continue,
press <Enter>. ";
        cin.get();
    }
};

int main(void)
{
    ART1 art1(3.0, 0.10);
    art1.run();
    return 0;
}

```

## RESULTADOS



```

Run  ia x
D:\Users\chama\Desktop\ia\cmake-bu
|
Finally, 8 clusters were found.

Cluster  1 0 0 0 0 0 1 0 1 0
includes
Sample  0: 1 0 0 0 0 1 1 0 1 1
Sample  1: 1 0 1 1 0 0 1 0 1 1
Sample  9: 1 0 0 1 0 0 1 1 1 0
Sample 21: 1 0 0 0 1 0 1 0 1 0

Cluster  1 0 0 1 1 0 0 0 1 0
includes
Sample  2: 1 0 1 1 1 0 0 1 1 0
Sample  3: 1 0 1 1 1 1 0 1 1 1
Sample 10: 1 1 1 1 1 1 0 0 1 0
Sample 11: 1 0 0 1 1 0 0 0 1 1
Sample 17: 1 0 1 1 1 1 1 0 1 0

Cluster  1 0 1 0 0 1 1 0 0 0
includes
Sample  4: 1 0 1 1 0 1 1 1 0 0
Sample  6: 1 0 1 0 1 1 1 1 0 1
Sample  7: 1 1 1 0 1 1 1 1 0 0
Sample 16: 1 0 1 0 0 1 1 0 0 1

Cluster  1 1 1 1 1 1 1 1 1 1
includes
Sample  5: 1 1 1 1 1 1 1 1 1 1

Cluster  0 0 0 1 0 1 1 0 0 0

```

```

Sample 4: 1 0 1 1 0 1 1 1 0 0
Sample 6: 1 0 1 0 1 1 1 1 0 1
Sample 7: 1 1 1 0 1 1 1 1 0 0
Sample 16: 1 0 1 0 0 1 1 0 0 1

Cluster 1 1 1 1 1 1 1 1 1 1
includes
Sample 5: 1 1 1 1 1 1 1 1 1 1

Cluster 0 0 0 1 0 1 1 0 0 0
includes
Sample 8: 1 0 0 1 1 1 1 0 0 0
Sample 13: 0 0 0 1 0 1 1 0 0 1
Sample 19: 1 1 0 1 0 1 1 0 0 1

Cluster 1 0 1 0 1 1 0 0 1 1
includes
Sample 12: 1 1 1 0 1 1 0 1 1 1
Sample 15: 1 0 1 0 1 1 1 0 1 1

Cluster 0 1 1 1 0 0 1 0 1 1
includes
Sample 14: 0 1 1 1 0 0 1 1 1 1
Sample 20: 1 1 1 1 0 0 1 0 1 1

Cluster 1 1 0 1 1 1 1 0 1 1
includes
Sample 18: 1 1 0 1 1 1 1 0 1 1

```

Como podemos observar habiendo hecho los cambios, metiendo las funciones de los prototipos dentro de la clase, así como los atributos de los vectores y matrices, tenemos los mismos resultados que el código original expuesto por el profesor en clase como se puede ver en la introducción, por lo que podemos confirmar que completamos el primer punto y funciona.

Pasando al segundo punto, al punto de las recomendaciones, tenemos lo siguiente, ya que hicimos un filtro donde hayan 4 o mas personas en el grupo, vemos los gustos que tienen cada uno

Revisando los resultados, en el clúster 0 tenemos 4 personas, pero no hay recomendaciones ya que no hay alguna persona que particularmente no posea un gusto que todos los demás sí.

```

Cluster 1 0 0 0 0 0 1 0 1 0
includes
Sample 0: 1 0 0 0 0 1 1 0 1 1
Sample 1: 1 0 1 1 0 0 1 0 1 1
Sample 9: 1 0 0 1 0 0 1 1 1 0
Sample 21: 1 0 0 0 1 0 1 0 1 0

Cluster 0 has 4 elements. Generating recommendations...

```

Pasando al segundo grupo con 4 o más personas tenemos el siguiente.

```

Cluster 1 0 0 1 1 0 0 0 1 0
includes
Sample 2: 1 0 1 1 1 0 0 1 1 0
Sample 3: 1 0 1 1 1 1 0 1 1 1
Sample 10: 1 1 1 1 1 1 0 0 1 0
Sample 11: 1 0 0 1 1 0 0 0 1 1
Sample 17: 1 0 1 1 1 1 1 0 1 0

Cluster 1 has 5 elements. Generating recommendations...
Sample 11 might consider feature 2 as all other members in the cluster have it.

```

Y podemos confirmar que para la persona 11, si hay algún gusto que no tenga, pero los demás si, así que hace la recomendación

Y por último, para el último grupo con 4 o mas personas tenemos el siguiente

```

Cluster 1 0 1 0 0 1 1 0 0 0
includes
Sample 4: 1 0 1 1 0 1 1 1 0 0
Sample 6: 1 0 1 0 1 1 1 1 0 1
Sample 7: 1 1 1 0 1 1 1 1 0 0
Sample 16: 1 0 1 0 0 1 1 0 0 1

```

```
Cluster 2 has 4 elements. Generating recommendations...  
Sample 16 might consider feature 7 as all other members in the cluster have it.
```

Podemos confirmar en este caso que la persona 16 tiene un gusto faltante que todos los demás si tienen, por lo que podemos confirmar que el código de recomendaciones funciona correctamente.

## ANALISIS

Al ejecutar el programa modificado, se observa que, al finalizar la iteración del algoritmo ART1, se generan recomendaciones para aquellas agrupaciones que tienen al menos 4 elementos. Las recomendaciones se presentan de forma clara y detallada, indicando las características específicas que los miembros de la agrupación podrían considerar.

La información sobre qué miembros ya consideran ciertas características proporciona una base sólida para que los usuarios tomen decisiones informadas sobre qué aspectos podrían ser beneficiosos explorar dentro de su agrupación.

## DISCUSION

La introducción de esta funcionalidad de sugerencias y recomendaciones añade valor al algoritmo ART1, convirtiéndolo en una herramienta más completa y orientada a la toma de decisiones. La capacidad de sugerir características específicas para considerar fortalece la utilidad del algoritmo en escenarios donde la interpretación de los resultados es crucial.

La modificación realizada no solo mejora la capacidad del algoritmo para agrupar elementos, sino que también proporciona una funcionalidad adicional que puede tener aplicaciones prácticas en la identificación de patrones y tendencias dentro de los conjuntos de datos analizados.